*Горшков Алексей Олегович*

*Группа 4212*

**Отчет о выполнении лабораторной работы №6 «Работа с растровыми изображениями в OpenGL»**

1. **Задание:** 
   1. Осуществить чтение/запись изображения в формате с типом компоненты цвета *К* (табл. 7) и форматом данных *Ф* (табл. 8). Реализовать вызов этой функции через подменю.
   2. Осуществить масштабирование изображения согласно заданию *М* (табл. 9). Добавить соответствующий пункт меню.
   3. Осуществить отображение только тех компонент, которые указаны в задании *Т* (табл. 10) Добавить соответствующий пункт меню.
   4. Осуществить инверсию компонент цвета, которые указаны в задании *Т* (табл. 10). Добавить соответствующий пункт меню.

Варианты заданий указаны в табл. 11. *Вариант 11 (см. ниже).*

Табл. 7.

|  |  |
| --- | --- |
| **Номер задания** | **Формат компонентов цвета** |
| К1 | Красный компонент цвета с сопутствующими зеленым и синим компонентами |
| К2 | Красный компонент цвета с сопутствующими зеленым, синим и альфа-компонентами |
| К3 | Синий компонент цвета с сопутствующими зеленым и красным компонентами |
| К4 | Синий компонент цвета с сопутствующими зеленым, красным и альфа-компонентами |
| К5 | Компонент яркости |
| К6 | Компонент яркости с сопутствующим альфа- компонентом |

Табл. 8.

|  |  |
| --- | --- |
| **Номер задания** | **Тип данных** |
| Ф1 | 8-разрядное целое число без знака |
| Ф2 | 8-разрядное целое число со знаком |
| Ф3 | 16-разрядное целое число без знака |
| Ф4 | 16-разрядное целое число со знаком |
| Ф5 | 32-разрядное целое число без знака |
| Ф6 | 32-разрядное целое число со знаком |
| Ф7 | Число с плавающей запятой одинарной точности |

Табл. 9.

|  |  |
| --- | --- |
| **Номер задания** | **Текст задания** |
| М1 | Отразить изображение относительно оси *Х* |
| М2 | Отразить изображение относительно оси *Y* |
| М3 | Отразить изображение относительно точки (0;0) |
| М4 | Увеличить изображение в 2 раза |
| М5 | Уменьшить изображение в 2 раза |

Табл. 10.

|  |  |
| --- | --- |
| **Номер задания** | **Компоненты цвета для отображения** |
| T1 | R |
| Т2 | G |
| Т3 | B |
| Т4 | R, G |
| Т5 | B, G |
| Т6 | R, B |
| Т7 | B, G |

Табл. 11.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Номер варианта** | **К** | **М** | **Т** | **Ф** |
| 1 | К1 | М1 | T6 | Ф6 |
| 2 | К2 | М2 | Т5 | Ф2 |
| 3 | К3 | М3 | Т4 | Ф3 |
| 4 | К4 | М4 | Т3 | Ф4 |
| 5 | К5 | М5 | Т2 | Ф5 |
| 6 | К6 | М1 | Т1 | Ф6 |
| 7 | К1 | М2 | T1 | Ф7 |
| 8 | К2 | М3 | Т2 | Ф1 |
| 9 | К3 | М4 | Т3 | Ф2 |
| 10 | К4 | М5 | Т4 | Ф3 |
| 11 | К5 | М1 | Т5 | Ф4 |
| 12 | К6 | М2 | Т6 | Ф5 |

1. **Листинг программы:**

#include <gl\glut.h>

#include <stdio.h>

#include <iostream>

#include <C:\\Program Files\\Microsoft Visual Studio\\2022\\Community\\VC\\Tools\\MSVC\\14.35.32215\\include\\GL\\glaux.h>

#pragma comment(lib, "C:\\Program Files\\Microsoft Visual Studio\\2022\\Community\\VC\\Tools\\MSVC\\14.35.32215\\lib\\x64\\GLAUX.lib")

#pragma comment(lib, "legacy\_stdio\_definitions.lib")

/\*Структура для хранения заголовка файла изображения\*/

struct Zagolovok

{

GLint shirina{}; /\*Ширина\*/

GLint vysota{}; /\*Высота\*/

GLenum formatCveta{}; /\*Формат представления цвета\*/

GLenum formatKomponenty{}; /\*Формат данных компоненты цвета\*/

int kol\_voKomponent{}; /\*Количество компонент цвета\*/

};

/\*Структура для хранения изображения\*/

struct Izobrajenie

{

unsigned char\* pikseli{};

Zagolovok zagolovok{};

};

Izobrajenie\* izobr{};//Текущее изображение.

Izobrajenie\* izobr\_bmp{};//Изображение из файла .bmp.

Izobrajenie\* izobr\_gb{};//Изображение с зеленым и синим компонентами.

AUX\_RGBImageRec\* pImage = NULL; //Изображение AUX\_RGB.

bool mirror\_image\_X{};//Переменная для учета отзеркаленности изображения от-но оси X.

static GLint rejim{-1};//Переменная для хранения режима рисования.

//Должным образом обновляем флаги в ответ на выбор позиции из меню.

void ObrabotkaMenu(int punktMenu)

{

//Меняем индекс режима визуализации на индекс, соответствующий позиции меню

rejim = punktMenu;

//Активизируем перерисовывание изображения glutPostRedisplay().

glutPostRedisplay();

}

void Pererisovka(void)

{

GLint iViewport[4]{};

//GLbyte \*pModifiedBytes = NULL.

GLfloat invertMap[256]{};//Инверсная карта цветов.

GLfloat defaultMap[256]{};//Стандартная карта цветов.

GLint i{};

//Очищаем окно текущим цветом очистки.

glClear(GL\_COLOR\_BUFFER\_BIT);

//Текущее растровое положение всегда соответствует левому нижнему углу окна.

glRasterPos2i(0, 0);

//В зависимости от индекса режима визуализации выполняются необходимые операции с изображением.

switch (rejim)

{

case 0: /\*Загрузка изображения из файла \*.bmp\*/

if (!izobr)

{

pImage = auxDIBImageLoad(L"space.bmp");

izobr\_bmp = (Izobrajenie\*)malloc(sizeof(Izobrajenie));

if (izobr\_bmp)

{

izobr\_bmp->zagolovok.shirina = pImage->sizeX\*3/2;

izobr\_bmp->zagolovok.vysota = pImage->sizeY;

izobr\_bmp->pikseli = pImage->data;

izobr\_bmp->zagolovok.formatCveta = GL\_LUMINANCE;

izobr\_bmp->zagolovok.formatKomponenty = GL\_SHORT;

izobr\_bmp->zagolovok.kol\_voKomponent = 1;

izobr = izobr\_bmp;

}

else

{

std::cout << "Не удалось выделить динамическую память из кучи! Куча переполнена." << std::endl;

free(izobr);

free(izobr\_bmp);

free(pImage);

exit(-1);

}

}

break;

case 1: /\*Очистка экрана\*/

// Сброс преобразованного изображения до первоначального с освобождением соответствующей динамической памяти

if (izobr)

{

defaultMap[0] = 0.0f;

for (i = 1; i < 256; i++)

defaultMap[i] = 1.0f / 255.0f \* (GLfloat)i;

glPixelMapfv(GL\_PIXEL\_MAP\_R\_TO\_R, 255, defaultMap);

glPixelMapfv(GL\_PIXEL\_MAP\_G\_TO\_G, 255, defaultMap);

glPixelMapfv(GL\_PIXEL\_MAP\_B\_TO\_B, 255, defaultMap);

glPixelTransferi(GL\_MAP\_COLOR, GL\_TRUE);

glPixelZoom(1, 1);

free(izobr);

free(pImage);

izobr = nullptr;

mirror\_image\_X = false;

}

glClearColor(0.0f, 0.0f, 0.0f, 0.0f);

break;

case 2: /\*Отражение изображения относительно оси X\*/

if (izobr != nullptr)

mirror\_image\_X = true;

break;

case 3:/\*Отображение только компонент G и B\*/

if (izobr)

{

if (mirror\_image\_X)

glPixelZoom(1, 1);

//Вначале рисуем изображение в буфере цвета.

glDrawPixels(izobr->zagolovok.shirina, izobr->zagolovok.vysota, izobr->zagolovok.formatCveta, izobr->zagolovok.formatKomponenty, izobr->pikseli);

//Распределяем память для карты яркости.

izobr\_gb = (Izobrajenie\*)malloc(sizeof(Izobrajenie));

if (izobr\_gb != nullptr)

{

izobr\_gb->zagolovok.shirina = izobr->zagolovok.shirina;

izobr\_gb->zagolovok.vysota = izobr->zagolovok.vysota;

izobr\_gb->zagolovok.formatCveta = GL\_LUMINANCE;

izobr\_gb->zagolovok.formatKomponenty = GL\_SHORT;

izobr\_gb->zagolovok.kol\_voKomponent = 1;

izobr\_gb->pikseli = (unsigned char\*)malloc(izobr\_gb->zagolovok.shirina \* izobr\_gb->zagolovok.vysota \* izobr\_gb->zagolovok.kol\_voKomponent \* 2);

//Масштабируем цвета согласно заданному условию.

glPixelTransferf(GL\_RED\_SCALE, 0.0f);

glPixelTransferf(GL\_GREEN\_SCALE, 1.0f);

glPixelTransferf(GL\_BLUE\_SCALE, 1.0f);

//Считываем пиксели в буфер.

glReadPixels(0, 400, izobr\_gb->zagolovok.shirina, izobr\_gb->zagolovok.vysota, izobr\_gb->zagolovok.formatCveta, izobr\_gb->zagolovok.formatKomponenty, izobr\_gb->pikseli);

if (mirror\_image\_X)

glClear(GL\_COLOR\_BUFFER\_BIT);

izobr = izobr\_gb;

//Масштабирование цвета возвращается в норму.

glPixelTransferf(GL\_RED\_SCALE, 1.0f);

glPixelTransferf(GL\_GREEN\_SCALE, 1.0f);

glPixelTransferf(GL\_BLUE\_SCALE, 1.0f);

}

else

{

std::cout << "Не удалось выделить динамическую память из кучи! Куча переполнена." << std::endl;

free(izobr);

free(izobr\_gb);

free(pImage);

exit(-1);

}

}

break;

case 4: /\*Инверсия компонент G и B\*/

if (izobr)

{

defaultMap[0] = 0.0f;

invertMap[0] = 1.0f;

for (i = 1; i < 256; i++)

{

defaultMap[i] = 1.0f / 255.0f \* (GLfloat)i;

invertMap[i] = 1.0f - (1.0f / 255.0f \* (GLfloat)i);

}

glPixelMapfv(GL\_PIXEL\_MAP\_R\_TO\_R, 255, defaultMap);

glPixelMapfv(GL\_PIXEL\_MAP\_G\_TO\_G, 255, invertMap);

glPixelMapfv(GL\_PIXEL\_MAP\_B\_TO\_B, 255, invertMap);

glPixelTransferi(GL\_MAP\_COLOR, GL\_TRUE);

}

break;

case 5:/\*Выход\*/

std::cout << "Завершение программы . . ." << std::endl;

if (izobr != nullptr)

{

free(izobr);

free(pImage);

}

exit(0);

break;

default:

break;

}

if (rejim != 1 && izobr != nullptr)

//Рисуются пиксели

{

if (mirror\_image\_X)

glPixelZoom(1, -1);

glPixelStorei(GL\_UNPACK\_ALIGNMENT, 1);

glDrawPixels(izobr->zagolovok.shirina, izobr->zagolovok.vysota, izobr->zagolovok.formatCveta, izobr->zagolovok.formatKomponenty, izobr->pikseli);

}

//Переключает буферы

glutSwapBuffers();

}

void IzmenenieRazmera(int w, int h)

{

//Предотвращает деление на ноль, когда окно слишком маленькое

if (h == 0)

h = 1;

glViewport(0, 0, w, h);

//Система координат обновляется перед модификацией

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

}

//Точка входа основной программы

int main(int argc, char\* argv[])

{

setlocale(LC\_ALL, "Rus");

glutInitDisplayMode(GLUT\_RGB | GL\_DOUBLE);

glutInitWindowSize(1200, 800);

glutCreateWindow("OpenGL\_Lab6");

glutReshapeFunc(IzmenenieRazmera);

glutDisplayFunc(Pererisovka);

//Создается меню и добавляются опции выбора

GLint submenu = glutCreateMenu(ObrabotkaMenu);

glutAddMenuEntry("Чтение изображения", 0);

glutAddMenuEntry("Очистить экран", 1);

glutCreateMenu(ObrabotkaMenu);

glutAddSubMenu("Загрузить изображение .bmp", submenu);

glutAddMenuEntry("Отобразить изображение относительно оси X", 2);

glutAddMenuEntry("Отображение только компонент G и B", 3);

glutAddMenuEntry("Инверсия компонент G и B", 4);

glutAddMenuEntry("Выход", 5);

glutAttachMenu(GLUT\_RIGHT\_BUTTON);

gluOrtho2D(0, 512, -512, 512);

glClearColor(0.0f, 0.0f, 0.0f, 0.0f);

rejim = 1;

glutMainLoop();

// Освобождаем исходные данные изображений

free(izobr);

free(izobr\_bmp);

free(izobr\_gb);

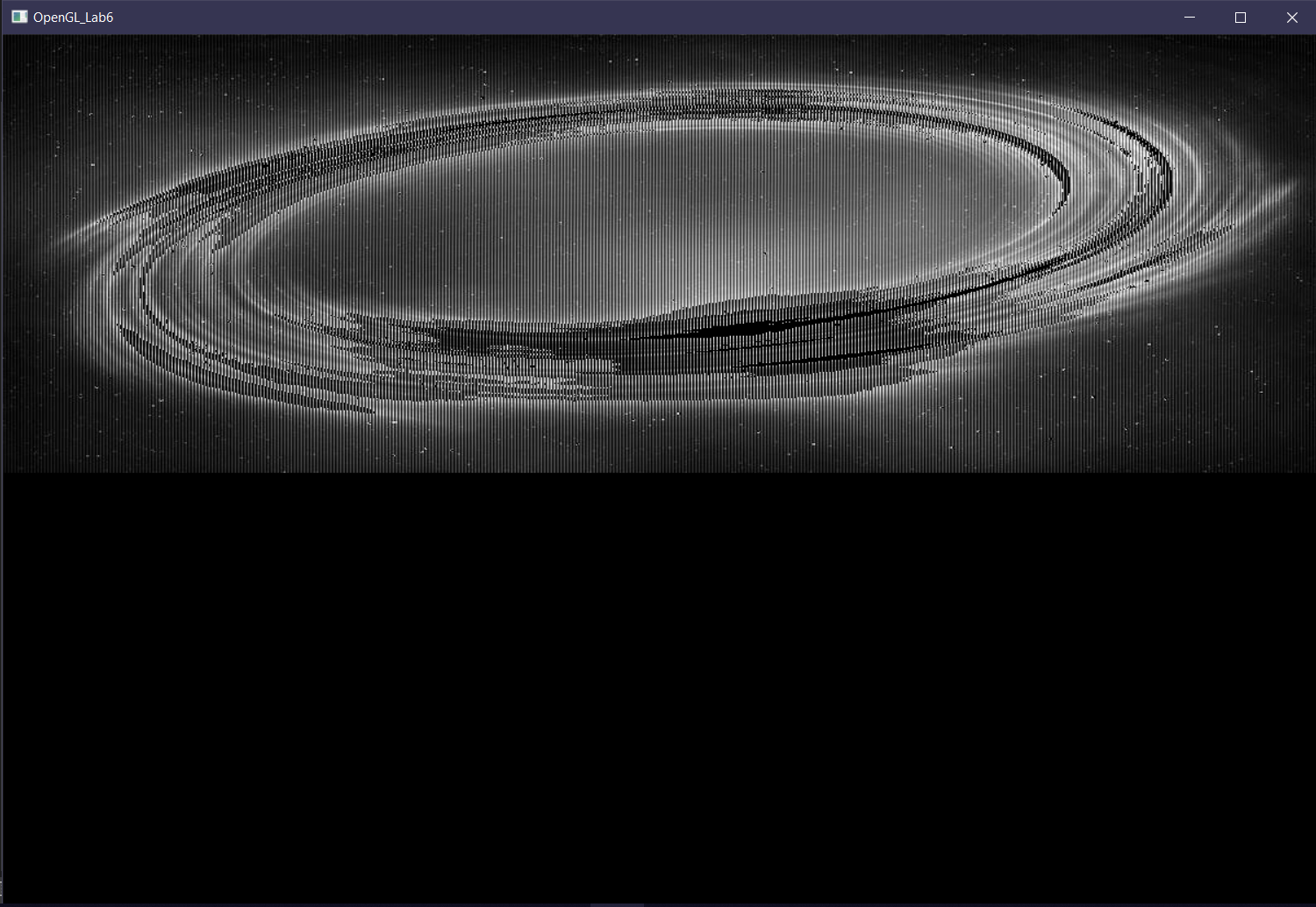
free(pImage);

return 0;

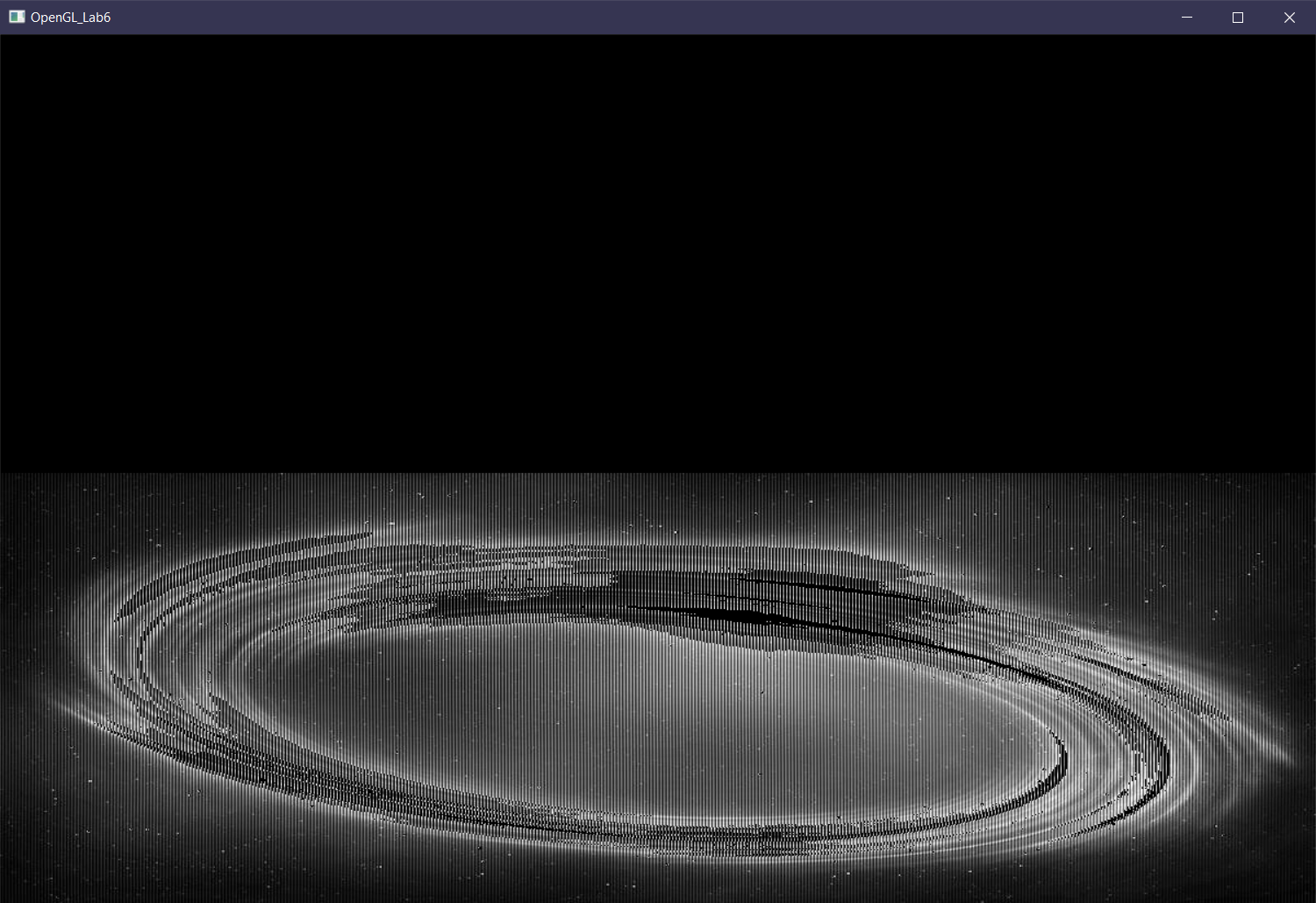
}

1. **Работа программы:**

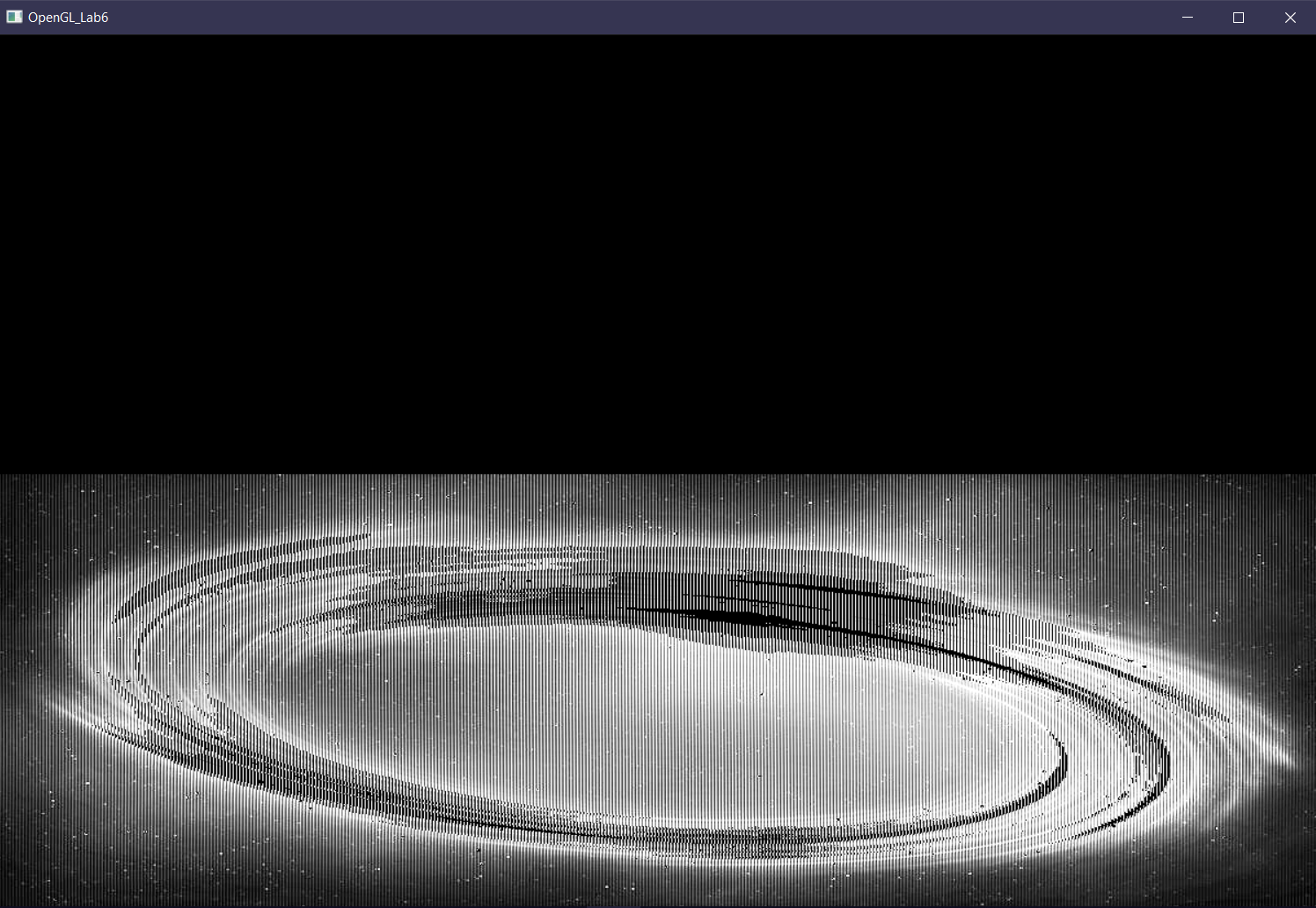
* Загрузить изображение .bpm -> чтение изображения:



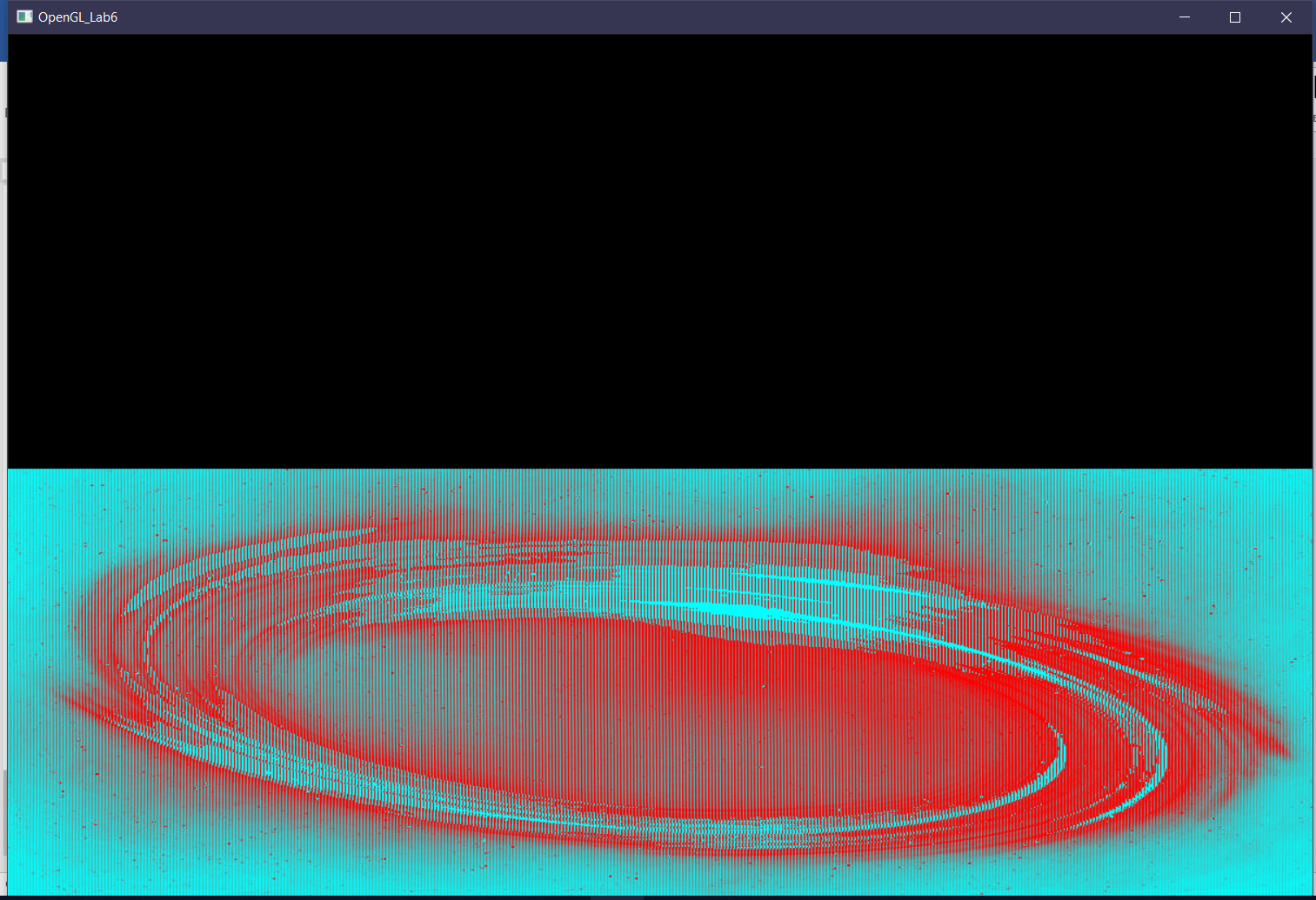
* Отображение относительно оси X:



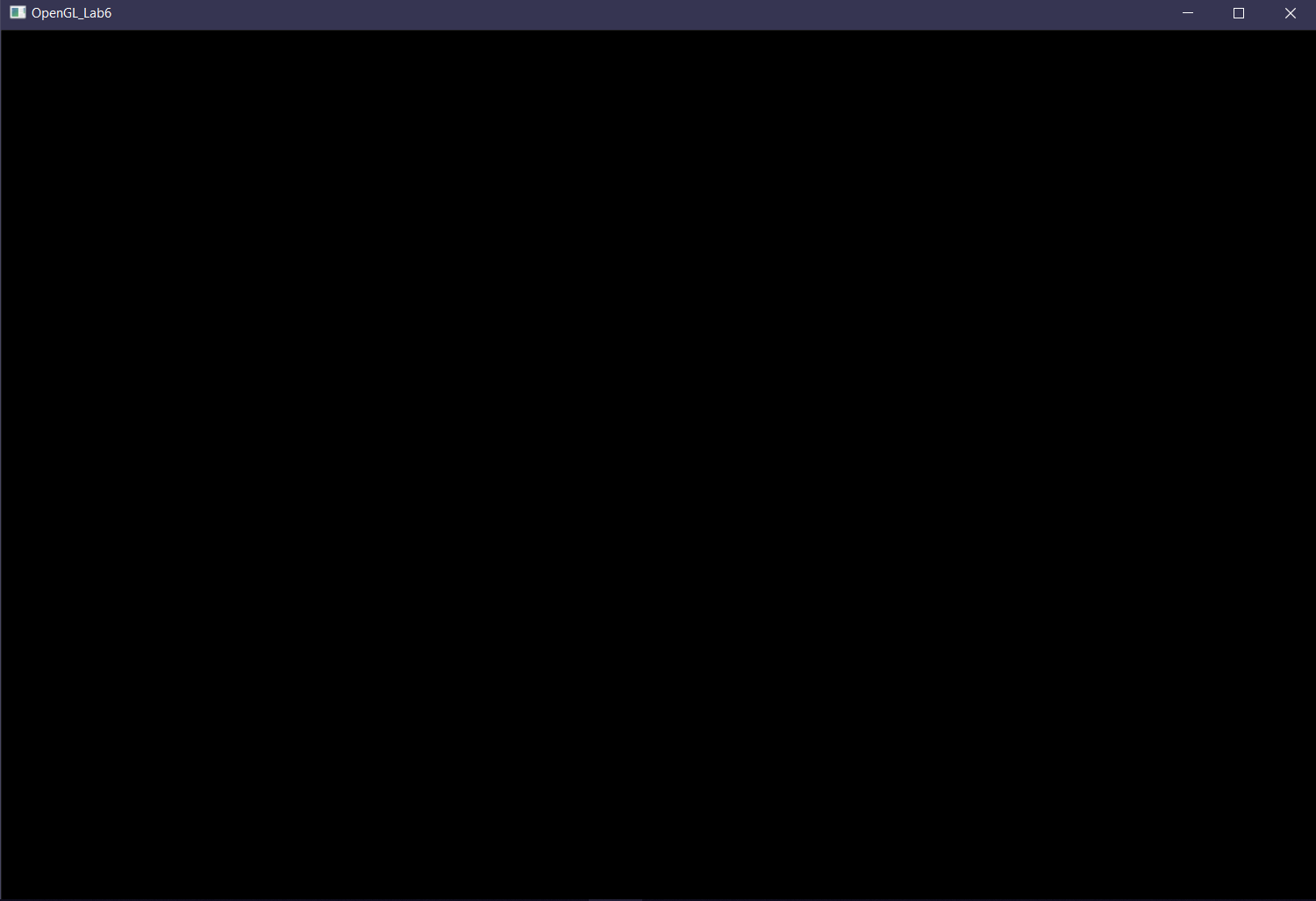
* Отображение только компонент G и B:



* Инверсия компонент G и B:



* Загрузить изображение .bpm –> Очистить экран:



* Выход:



Меню и подменю программы:

